**Natan Kołodziej**

**03.12.20**

**Sprawozdanie nr 4**

**Temat: Django i Django Rest framework**

**Teoria:**

Wstęp teoretyczna (jeżeli wspomniałem o tym w treści zadania).

Django Rest framework jest używany do tworzenia api na stronach opartych o django.

**Przebieg zadania:**

Część praktyczna (kod, screeny, opis działania funkcji, opis przebiegu zadania)

Project/urls.py

from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('',include('basic.urls'))  
]

projects/settings.py

from pathlib import Path  
  
# Build paths inside the project like this: BASE\_DIR / 'subdir'.  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = '6cydb(hx$8c+y8k)09t5pb7572!n2kz4\_unk\_4mt3lhay7x3=^'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'rest\_framework',  
 'basic',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'project.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'project.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}  
  
  
# Password validation  
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/3.1/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_L10N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/3.1/howto/static-files/  
  
STATIC\_URL = '/static/'

basic/admin.py

from .models import Case  
from django.contrib import admin  
  
# Register your models here.  
  
  
admin.site.register(Case)

basic/models.py

from django.db import models  
  
  
# Create your models here.  
class Case(models.Model):  
 producent = models.CharField(max\_length=25)  
 name = models.CharField(max\_length=25)  
 price = models.CharField(max\_length=5)  
  
 def \_\_str\_\_(self):  
 return self.producent

basic/serializers.py

from rest\_framework import serializers  
from .models import Case  
  
  
class CaseSerializer(serializers.ModelSerializer):  
 class Meta:  
 model= Case  
 fields = ['id', 'producent', 'name', 'price']

basic/urls.py

from django.urls import path  
from .views import case\_list  
urlpatterns = [  
 path('case/', case\_list),  
]

basic/views.py

from django.shortcuts import render  
from django.http import HttpResponse, JsonResponse  
from rest\_framework.parsers import JSONParser  
from .serializers import CaseSerializer  
from .models import Case  
# Create your views here.  
def case\_list(request):  
 if request.method=='GET':  
 cases= Case.object.all()  
 serializer= CaseSerializer(cases, many=True)  
 return JsonResponse(serializer.data, safe=False)  
  
 elif request.method=='POST':  
 data=JSONParser().parse(request)  
 serializer = CaseSerializer(data=data)  
  
 if serializer.is\_valid():  
 serializer.save()  
 return JsonResponse(serializer.data, status=201)  
 return JsonResponse(serializer.errors, status=400)

Zapoznanie się z dokumentacja django i django rest framework. Obejrzenie filmu na youtube. Bazując na nim i dokumentacji stworzenie aplikacji która ma możliwość z kontroli administratora na dodawanie, usuwanie i edytowanie obiektów case.